EXHIBIT D

EXHIBIT H

Expert Report – '890 Invalidity Chart
Exhibit [H]

Analysis of ARM for U.S. Patent 5,530,890

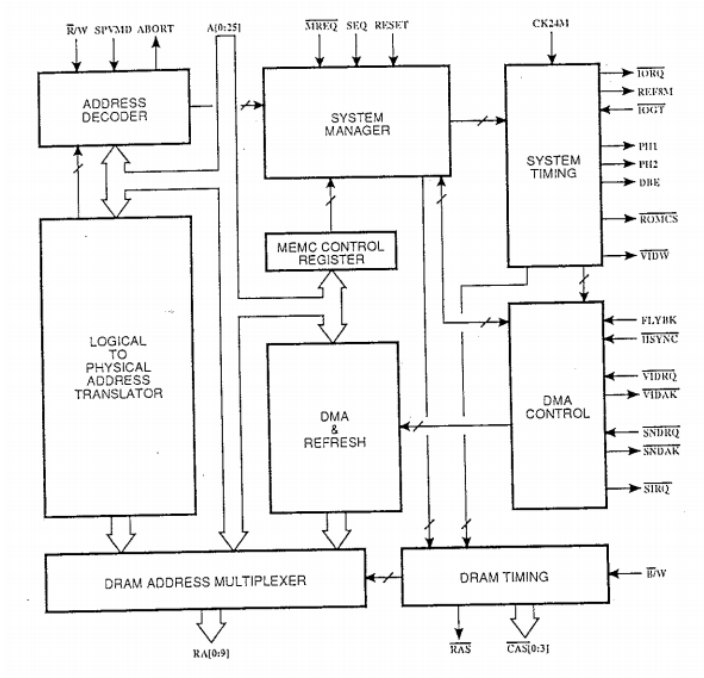
The ARM Architecture [1] anticipates claim 11 of the U.S. Patent No. 5,530,890 (the "'890 patent") under at least 35 U.S.C. §§ 102(a), 102(b), and/or 102(g), and renders obvious the '890 patent under 35 U.S.C. § 103 alone and/or in combination with other references.

Claim Language	
Claim 11	
A microprocessor, which comprises	To the extent that the preamble of Claim 11 is limiting, the ARM Architecture discloses this limitation. For example: The ARM (Acorn RISC Machine) is a general purpose 32-bit single-chip microprocessor. [HTCTP10217798] Further support is found in the ARM Assembly Language Programming Reference: In most microcomputer systems, the CPU occupies a single chip (integrated circuit), housed in a plastic or ceramic package. The ARM CPU is in a square package with 84 connectors around the sides. Section 1.4 describes in some detail the major elements of the ARM CPU. In this section we are more concerned with how it connects with the rest of the system. [HTCTP10217692]
a main central processing unit and	The ARM Architecture discloses this limitation. For example: The ARM (Acorn RISC Machine) is a general purpose 32-bit single-chip

[1]The claim chart contains exemplary citations to the portions of the 1987 ARM Datasheet that disclose each claim limitation. For additional support, I have also provided citations to a book titled "ARM Assembly Language Programming Reference" by Pete Cockerell. This reference was published in 1987 and describes the identical ARM system shown in the 1987 ARM Datasheet. Both of these references describe a single ARM architecture (hereinafter referred to as the "ARM Architecture").

Claim Language	
	microprocessor. [HTCTP10217798] Further support is found in the ARM Assembly Language Programming Reference: In most microcomputer systems, the CPU occupies a single chip (integrated circuit), housed in a plastic or ceramic package. The ARM CPU is in a square package with 84 connectors around the sides. Section 1.4 describes in some detail the major elements of the ARM CPU. In this section we are more concerned with how it connects with the rest of the system. [HTCTP10217692]
a separate direct memory access central processing unit in a single integrated circuit comprising said microprocessor,	The Court has construed "separate direct memory access central processing unit" as "a central processing unit that accesses memory and that fetches and executes instructions directly and separately of the main central processing unit." As discussed above, however, one skilled in the art will understand that a "separate direct memory access central processing unit" is dedicated to performing "direct memory access" operations. Otherwise, the term "direct memory access," or 'DMA,' in this limitation would be superfluous. This is confirmed in Figure 5 of the '890 patent, which shows a basic DMA architecture, compared with the complex main CPU architecture shown in Figure 2. Under the Court's construction, the ARM Architecture discloses or renders obvious this limitation. For example, the 1987 ARM Datasheet discloses the use of co-processor capability which requires addition of "further logic" to an "integrated memory controller": MEMC (VTI part number VL86C 110) is an integrated memory controller for ARM which incorporates an address translation system and generates all the critical system timing signals.… If Co-Processor capability is required, further logic must be added to modify the behaviour of MEMC. [HTCTP10217849]

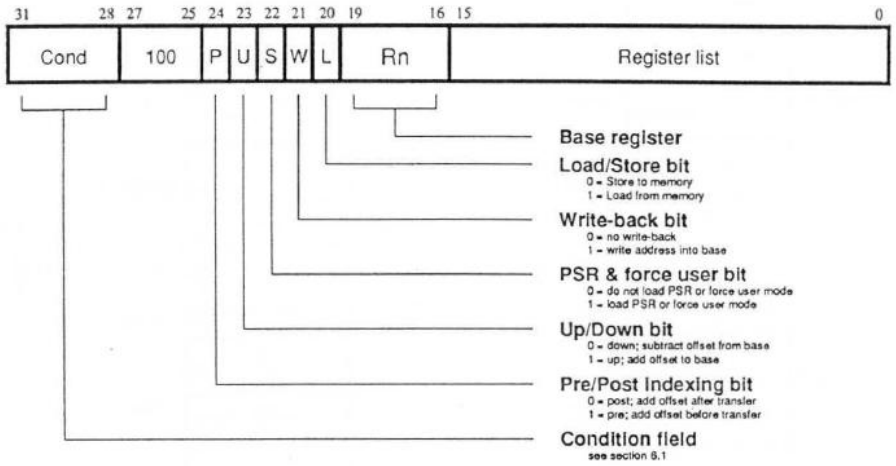
Expert Report – '890 Invalidity Chart
Exhibit [H]

Claim Language	
	The MEMC was a separate chip also developed in the mid 1980s that controlled DMA operations. The chip was another ACORN product like the ARM processor and was produced at about the same time. The MEMC Datasheet (also published in the 1980s), shown below, discloses a DMA controller that allowed for the addition of further logic to provide co-processor capability. [*See* HTCTP10218759 – HTCTP10218836].  [HTCTP10218765, MEMC Datasheet at 3] The Co-Processor could contain a queue of such instructions awaiting execution, and their execution can overlap other ARM activity allowing the Co-Processor and ARM to perform independent tasks in parallel. [HTCTP10217834, Datasheet at 37]

Claim Language	
	As explained in the ARM Datasheet, the ARM co-processor interface suggested that a DMA controller (such as the MEMC) using the co-processor interface could fetch and execute instructions from a queue. One of ordinary skill would understand that such an enhancement to the MEMC would provide the ability to fetch and execute instructions like a processor or a CPU. The motivation would be to provide enhanced video and communications capability. Although the 1987 ARM Datasheet describe the co-processors as external devices, it would have been obvious to one skilled in the art to include both the main processor and the co-processors on a single integrated circuit as the motivation to reduce size in microprocessor design had existed for a long time in 1987. TPL, in its infringement contentions, has mapped this claim limitation to cover any second processor residing on the same integrated circuit. [*See, e.g.*, Jan. 16, 2013 Infringement Contentions, Exhibit HTC B-1 at 3]. Dual processor systems were well known in the 1980s. For example, U.S. Patent No. 4,679,166 (the “’166 patent”), which claims priority to a Jan 17, 1983 application, discloses a dual processor system which allowed additional functionality such as video processing. A dual processor system in which one processor is dedicated to input/output tasks while the other is dedicated to high level language tasks when operating as a 16-bit machine. [HTCTP10218691,the ’166 patent, Abstract]

Claim Language	
	[HTCTP10218692, the ’166 patent, Fig 1] It would have been obvious to one skilled in the art to combine the ARM Architecture with the ’166 patent to have a MEMC co-processor on the same chip. The motivation for such a combination would have been to produce a more flexible, single chip product that provides more programmability to allow functions such as enhanced video and graphics processing. Under HTC’s proposed construction of the “separate DMA CPU,” which is “a separate CPU that fetches and executes instructions for performing direct memory access without using the main CPU,” the ARM Architecture renders obvious this limitation. For example, as explained above, an enhanced MEMC co-processor would fetch and execute DMA instructions. It would have been obvious to put both the main

Claim Language	
	processor and the co-processor on a single chip. The motivation to do this would be to simplify the hardware design and architecture of the system. As shown by the ’166 patent above, the use of additional processors on a single chip to handle discrete operations was well known. One of ordinary skill would be similarly motivated to combine the ’166 patent’s concept with the ARM Architecture to construct the MEMC co-processor on a single chip to be dedicated to performing DMA operations, which would eliminate the need to have DMA controllers.
said main central processing unit having an arithmetic logic unit,	The ARM Architecture discloses this limitation. For example, as shown in the ARM Datasheet:

Claim Language	
	[HTCTP10217826, ARM Datasheet at 29] Furthermore, one skilled in the art would understand that mode and instruction registers exist in an ARM. All conventional processors had mode registers as of the priority date. (*See, e.g.*, U.S. Patent Nos. 4,849,875 (7: 17-29) [HTCTP10218733-HTCTP10218758]; 4,847,757 (5:26-34) [HTCTP10218712-HTCTP10218732]; 4,835,733 (2:48-56) [HTCTP10218700-HTCTP10218711]; 4,016,545 (9:61-10-2) [HTCTP10218637-HTCTP10218678]; 3,930,688 (7:1-19) [HTCTP10218618-HTCTP10218636])
said stack pointer pointing into said first push down stack,	The ARM Architecture discloses or renders obvious this limitation. For example, as shown below in the ARM Datasheet:  [HTCTP 10217826] Block data transfer instructions are used to load (LDM) or store (STM) any subset of the currently visible registers. They support all possible stacking modes, maintaining full or empty stacks which can grow up or down memory, and are very efficient instructions for saving or restoring context, or for moving large blocks of data around main memory.